Лабораторна робота №7

Тема: Реалізація транзакцій, блокування та резервування даних.

Мета: навчитися виконувати реалізацію транзакцій, блокування та резервування даних.

*Питання для самопідготовки:*

1. Транзакції.
2. Блокування таблиць.
3. Резервування бази даних.
4. Профілактична перевірка та відновлення таблиць.

Теоретичні відомості

1. Транзакції

***Транзакція*** - послідовність операторів SQL, що виконуються як єдина операція, яка не переривається іншими клієнтами. Поки відбувається робота з записами таблиці (їх оновлення або видалення), СУБД MySQL автоматично блокує доступ до них.

Підтримка транзакцій здійснюється тільки в таблицях ВDВ і InnoDB.

Транзакції об'єднують оператори в групу і гарантують, що всі операції всередині групи будуть виконані успішно. Якщо частина транзакції виконується зі збоєм, результати виконання всіх операторів транзакції до місця збою скасуються, приводячи базу даних до вигляду, в якому вона була до виконання транзакції.

За замовчуванням СУБД MySQL працює в режимі автоматичного завершення транзакцій, після виконання оператора оновлення даних, який модифікує таблицю, MySQL зберігає зміни на диску. Для об'єднання в транзакцію кількох операторів слід відключити цей режим. Це здійснюється за допомогою системної змінної ***AUTOCOMMIT***, значення якої встановлюється оператором ***SET AUTOCOMMIT = 0;***

Після відключення режиму автоматичного завершення транзакцій слід використовувати оператор ***COMMIT***, щоб зберігати зміни на диску, або ***ROLLBACK***, щоб скасувати зміни, виконані з моменту початку транзакції. Для включення режиму автоматичного завершення транзакцій, слід виконати оператор ***SET AUTOCOMMIT=1.***

Для включення режиму автоматичного завершення транзакцій тільки для окремої послідовності операторів, використовується оператор ***START TRANSACTION***.

Після виконання оператора ***START*** ***TRANSACTION*** режим автоматичного завершення транзакцій залишається включеним до явного завершення транзакції за допомогою оператора ***COMMIT*** або відкату транзакції за допомогою ***ROLLBACK***.

***Приклад механізму транзакцій.***

UPDATE user\_account SET allsum = allsum + 1000 WHERE id =‘1’;

UPDATE user\_account SET allsum = allsum - 1000 WHERE id =‘2’;

Це переказ грошей з особового рахунку клієнта з номером 2 на особовий рахунок клієнта з номером 1. Якщо перший запит успішно виконався, а другий з якихось причин (помилка бази даних, помилка на сервері і т.д.) - ні. Отримуємо ситуацію, яка загрожує грошовими втратами.

Для уникнення цього потрібно, щоб обидва запити виконувалися як одне ціле. І якщо виникла помилка в одному запиті, не були виконані інші. Для цього служить ***механізм транзакцій.***

Оператор, що відкриває транзакцію: ***"START TRANSACTION;"***. Після правильного виконання всіх запитів транзакцію можна або завершити внісши всі зміни в базу даних - ***"COMMIT;",*** або відкотити, повернувши все в початковий стан – ***“ROLLBACK".***

Слід врахувати що:

1. Транзакційний механізм підтримують тільки InnoDB і BDB. Тому всі таблиці з якими хочете працювати через транзакції слід переконвертувати у відповідний тип.

2. За замовчуванням MySQL працює в режимі autocommit. Це означає, що результати виконання будь-якого SQL - оператора, що змінює дані, будуть відразу зберігатися.

Режим ***autocommit*** можна відключити командою ***SET AUTOCOMMIT = 0***. При відключеному режимі ***autocommit*** кожну транзакцію треба явно завершувати операторами ***COMMIT/ROLLBACK.***

Таким чином, для того, щоб реалізувати однократну транзакцію, яка вирішує поставлену на початку статті проблему, необхідно виконати такий код:

*START TRANSACTION;*

*UPDATE user\_account*

*SET allsum = allsum + 1000 WHERE id =‘1’;*

*UPDATE user\_account*

*SET allsum = allsum - 1000 WHERE id =‘2’;*

*COMMIT;*

2. Блокування таблиць

Для таблиць типу MyІSAM використання транзакцій недоступне. Проте їх можна емулювати при допомогою операторів LOCK TABLES та UNLOCK TABLES. Ці оператори блокують всю таблицю, ніхто не може працювати з таблицями до тих пір, поки вони залишаються заблокованими. Оператор LOCK TABLES виконує блокування таблиць, a UNLOCK TABLES знімає блокування.

Усі таблиці, заблоковані в поточному з'єднанні, розблокуються при повторному виклику оператора LOCK TABLES.

Оператори LOCK TABLES і UNLOCK TABLES мають синоніми LOCK TABLE і UNLOCK TABLE, відповідно

*LOCK TABLES catalogs WRITE;*

*INSERT INTO catalogs VALUES (NULL, ‘Принтер');*

*INSERT INTO catalogs VALUES(NULL, ‘Різне');*

*UNLOCK TABLES;*

Лістинг демонструє блокування таблиці ***catalogs*** на час додавання даних в базу даних. Після оператора LOCK TABLES записується ім'я блокованої таблиці, при знятті блокування за допомогою оператора UNLOCK TABLES вказання імені таблиці не потрібне. Основна причина застосування блокування таблиці за допомогою LOCK TABLES - це збільшення швидкості оновлення таблиць і додавання великих обсягів даних.

При використанні блокувань можна явно вказати тип блокування - на читання (READ) або на запис (WRITE).

*LOCK TABLES catalogs WRITE;*

*INSERT INTO catalogs VALUES(NULL,'Периферія')*

*INSERT INTO catalogs VALUES(NULL,'Різне');*

*UNLOCK TABLES catalogs;*

*LOCK TABLES catalogs READ;*

*INSERT INTO catalogs VALUES (NULL,'Периферія')*

*INSERT INTO catalogs VALUES(NULL,‘Різне');*

*UNLOCK TABLES;*

 Різниця між блокуваннями (READ) і (WRITE) полягає в тому, що клієнт, який встановив блокування на читання, і інші клієнти можуть тільки читати з таблиці дані. При блокуванні на запис (WRITE) встановивший його клієнт може як вносити записи в таблицю, так і читати, в той час як доступ інших клієнтів до таблиці блокується. Причому всі інші клієнти очікують, коли блокування буде скасоване оператором UNLOCK TABLES.

Один оператор LOCK TABLES може блокувати декілька таблиць, рекомендується блокувати всі таблиці, які беруть участь у запитах всередині блокування.

*LOCK TABLES catalogs WRITE, products WRITE;*

*SELECT \* FROM catalogs, products*

*WHERE products.id\_catalog = catalogs.id\_catalogs*

*INSERT INTO catalogs VALUES(NULL,‘Різне');*

*UNLOCK TABLES catalogs;*

3. Резервування бази даних

3.1.Повне резервування

Для повного резервного копіювання баз даних призначена утиліта ***mysqldump***.

Для запуску утиліти з командного рядка Windows слід виконати команду

***mysqldump -u <Ім'я користувача> -p***

***[Опціональні параметри]***

***< Копійовані бази даних і таблиці>*** ***> < Шлях та ім'я результуючого файлу>***

Після появи запрошення ***Enter password*** (Введіть пароль) введіть пароль користувача.

Вибір опціональних параметрів утиліти залежить від типу резервуються таблиць.

* Для резервування таблиць InnoDB слід вказати параметр-***single -transaction.***
* При резервуванні таблиць MyISAM слід заборонити користувачам зміну даних, щоб уникнути їх неузгодженості. Для цього необхідно вказати параметр ***-lock -all -tables*** або ***-lock -tables***.
* Копійовані об'єкти можна задати одним із способів:
* ***-all –databases***
* необхідне копіювання всіх баз даних з сервера MySQL.
* ***-databases <Ім'я бази даних1> <Ім'я бази даних2>...*** необхідно скопіювати перераховані бази даних.
* ***<Ім'я бази даних> <Ім'я таблиці1> <Ім'я таблиці2>...*** необхідно скопіювати перераховані таблиці вказаної бази даних.

Наприклад, команда

***mysqldump -u root -p -single -transaction -flush -logs***

***-Databases SalesDept FinanceDept > "C:\data\full\_backup.sql"***

виконує резервне копіювання баз даних ***SalesDept*** і ***FinanceDept*** у файл ***full\_backup.sql***, що знаходиться в папці C:\data, а команда

***mysqldump -u root -p -lock -tables -flush -logs***

***mysql user db tables\_priv columns\_priv > “C:\data\users.sql”***

виконує резервне копіювання таблиць ***user, db, tables\_priv*** і ***columns\_priv*** системної бази даних ***mysql*** в файл ***users.sql***, що знаходиться в папці C:\data. Таблиці в базі даних ***mysql*** мають тип MyISAM, тому при резервуванні ми вказали параметр***-lock -tables.***

3.2 Відновлення даних

Щоб відновити базу даних з файлу, що містить повну резервну копію, слід виконати команду з командного рядка Windows

***mysql -u root -p [<Ім'я бази даних>] < <Шлях та ім'я файлу>***

Після появи запрошення Enter password ввести пароль користувача root.

Якщо резервна копія була створена командою

***mysqldump -u root -p -single -transaction -flush -logs***

***-Databases SalesDept FinanceDept > “ C: \data full\_backup.sql”***

то відновити бази даних SalesDept і FinanceDept можна командою

***mysql -u root -p < “C:\data full\_backup.sql”***

Якщо резервна копія була створена за допомогою команди

***mysqldump -u root -p -lock -tables -flush -logs***

***mysql user db tables\_priv columns\_priv > “C:\data \users.sql”***

то при відновленні необхідно вказати ім'я бази даних, в яку будуть поміщені відтворені таблиці user, db, tables\_priv і columns\_priv:

***mysql -u root -p mysql < “C:\data \users.sql”***

4. Профілактична перевірка та відновлення таблиць

4.1Профілактична перевірка таблиць

Для перевірки таблиць виконати команду

***CHECK TABLE <Список таблиць>;***

Команда CHECK TABLE відображає результат перевірки таблиць. Наприклад, щоб отримати інформацію про стан таблиць db і user системної бази даних mysql, виконайте команду

***CHECK TABLE mysql.db, mysql.user;***

Якщо в стовпці ***Msg\_text*** (текст повідомлення) міститься значення, відмінне від ***OK*** або ***Table is already up to date*** (Таблиця вже перевірена), то таблиця пошкоджена.

4.2 Відновлення таблиць

Для відновлення таблиці слід виконати такі дії.

1. Вибрати команду ***REPAIR TABLE <Ім'я таблиці> QUICK;***

Результат виконання команди REPAIR TABLE аналогічний результату виконання команди CHECK TABLE. Якщо в останньому рядку в стовпці Msg\_text (текст повідомлення) вказано значення OK, то таблиця успішно відновлена. В іншому випадку перейдіть до наступного пункту.

2. Скопіювати файл ***<Ім'я таблиці>. MYD*** з папки ***<Коренева папка MySQL >\data\<Ім'я бази даних>*** в будь-яку резервну папку, тому щоспроби відновлення можуть пошкодити дані, які містяться в цьому файлі.

3. Виконати команду

***REPAIR TABLE <Ім'я таблиці>;***

Якщо і ця команда не допомогла відновити таблицю, виконати команду

***REPAIR TABLE <Ім'я таблиці> EXTENDED;***

Якщо знову не вийшло виправити ушкодження, виконати команду

***REPAIR TABLE <Ім'я таблиці> USE\_FRM;***

Параметр ***USE\_FRM*** повинен використовуватися тільки в тому випадку, якщо попередні дії не дали потрібного результату.

Якщо таблиця так і не була відновлена​​, перейти до наступного пункту.

1. Відкрити файл з повною резервною копією бази даних. Знайти у ньому SQL - команду CREATE TABLE для тієї таблиці, яку потрібно відновити. За допомогою цієї команди створити точно таку ж таблицю в іншій базі даних. Потім перемістити файли ***<Ім'я таблиці>. MYI*** і ***<Ім'я таблиці>.frm*** з папки <Коренева папка MySQL > \data\<Ім'я іншої бази даних> в папку ***<Коренева папка MySQL > \data\<Ім'я вихідної бази даних>.*** Повторити дії, описані в п. 3.

Завдання для студентів

1. Виконати реалізацію механізму транзакцій.
2. Виконати реалізацію блокування таблиць на читання та запис.
3. Виконати реалізацію розблокування таблиць бази даних.
4. Виконати повне резервування бази даних.
5. Виконати відновлення бази даних.
6. Виконати профілактичну перевірку бази даних.
7. Виконати відновлення таблиць бази даних.

Контрольні питання

1. Як виконується реалізація механізму транзакцій?
2. Як виконується блокування таблиць на читання та запис?
3. Як виконується розблокування таблиць бази даних?
4. Як виконується повне резервування бази даних?
5. Як виконується відновлення бази даних?
6. Як виконати профілактичну перевірку бази даних?
7. Як виконується відновлення таблиць бази даних?